CALIFORNIA INSTITUTE OF TECHNOLOGY

Computer Science

5043:TM:82

A Formal Derivation of Array Implementations of FFT Algorithms

by

Lennart Johnsson and Danny Cohen

Presented at
USC Workshop on VLSI and Modern Signal Processing
(sponsored by ONR)
November 1982

The research described in this paper was sponsored by the
Defense Advanced Research Projects Agency, ARPA Order number 3771,
and monitored by the
Office of Naval Research
under contract number N00014-79-C-0597

A FORMAL DERIVATION OF ARRAY IMPLEMENTATIONS OF FFT ALGORITHMS

Lennart Johnsson
Computer Science
California Institute of Technology
Pasadena, California 91125

Danny Cohen
Information Sciences Institute
University of Southern California
Marina Del Rey, California 90291

ABSTRACT

Fast Fourier Transform, FFT, algorithms are interesting for direct hardware implementation in VLSI. The description of FFT algorithms is typically made either in terms of graphs illustrating the dependency between different data elements or in terms of mathematical expressions without any notion of how the computations are implemented in space or time. Expressions in the notation used in this paper can be given an interpretation in the implementation domain. The notation is in this paper used to derive a description of array implementations of decimation-in-frequency and decimation-in-time FFT algorithms. Correctness of the implementations is guaranteed by way of derivation.

INTRODUCTION

The notation and the methodology used in this paper explicitly model storage, arithmetic, and logic operations. The notation has great resemblance with the notation used in the treatment of discrete-time systems and difference equations. The form used here was first described and given an interpretation in terms of computational networks in Cohen, [Cohen 78], and has later been applied to two-dimensional networks, [Weiser, Davis 80], [Johnsson et. al. 81], [Weiser, Davis 81], and extended to include explicit control [Cohen, Tyree 79], [Johnsson, Cohen 81a], and [Johnsson, Cohen 81b].

The correspondence between expressions in the mathematical notation and computational networks makes possible the use of formal symbolic transformations in the synthesis and analysis of concurrent algorithms. It is possible to proceed from a description of an algorithm in standard mathematical notation to a description of an implementation of it in space and time without leaving the domain of a mathematical notation.

The traditional mathematical notation is function-oriented, with no regard to issues related to distributing computations in space and time. By introducing the notion of storage and explicitly modeling control of computational devices a wide class of computational networks can be treated formally.

Without using a storage operator it is possible to model only combinational state-free networks composed of only memory-free functional units. However, the use of a storage operator allows for the mathematical modeling of the behavior of any computational network using both combinational logic and storage.

A major concern in design is the trade-off between time and space. A full instantiation of an algorithm in space is a fully concurrent implementation, whereas a full instantiation in time is a fully sequential implementation.

DEFINITIONS

Following [Cohen 78], the operator Z is defined by:

$$Zx(j) = x(j-1)$$

Let $Z^k = ZZ^{k-1}$

Then $Z^k x(j) = x(j-k)$

Define Z^{-1} to be the inverse of Z such that

$ZZ^{-1} = Z^{-1}Z = Z^0 = I$, where I is the identity operator.

If {x} is a sequence of data elements such that element x(j) precedes x(j+1), then j can be interpreted as time, Z as a delay, and its inverse as a prediction.

The operator Z is a model of a storage cell such as a flip-flop or a shift register cell. Z^N models a shift register of length N.

The operator Z commutes with time-independent functions:

$$ZF(U) = F(ZU) \qquad (1)$$

For example, if U={u,v} then $ZF(u,v) = F(Zu,Zv)$

Figure 1 shows a graphical representation of this commutative distributive property.

Figure 1: $ZF(u,v) = F(Zu,Zv)$

If C is a constant, then ZC=C and the following holds:

$$Z(CX) = (ZC)(ZX) = C(ZX) = CZX$$

Hence, as operators Z and C commute.

A binary multiplexor has an output signal M(X,Y,U) that is equal to one or the other of its two inputs X and Y according to a control signal U:

$$M(X,Y,U) = \left\{ \begin{array}{l} X, \text{ if } U=1 \\ \\ Y, \text{ if } U=0 \end{array} \right\} = UX + U'Y \qquad (2)$$

A graphical representation of the multiplexor is shown in Figure 2.

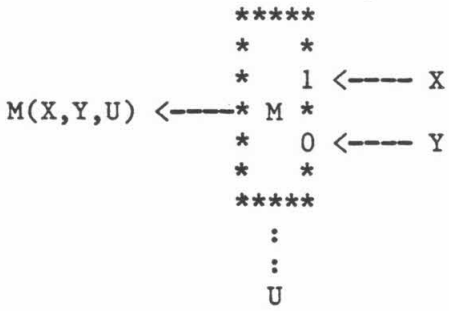

Figure 2: Graphic representation of a multiplexer

The expression UX+U'Y is formally a sum of two products of which at least one is equal to zero. This notation is convenient in our formal derivations. By the way the control signal is introduced and defined, the selection mechanism requires no special treatment in our formalism.

When there is a need to emphasize implementation aspects we use the "#"-sign instead of the "+"-sign (e.g., UX#U'Y) to indicate the combination of two entities, of which only one is actually valid (enabled, non-zero, etc).

Throughout this note upper case symbols (like X) denote signals and operators, whereas lower case symbols (such as x) denote specific values.

THE DISCRETE FOURIER TRANSFORM

The Discrete Fourier Transform, DFT, of order N is defined by

$$y(m,k) = \sum_{j=0}^{N-1} w^{mj} x(k+j) \qquad m=0,1,\ldots,N-1 \qquad (3)$$

where $w = w_N = e^{-2*\pi*i/N}$

One of the main ideas behind FFT algorithms, [Cooley, Tukey 65], is exploiting the following property of the w's:

$$w_N^{mj} = \left\{ \begin{array}{ll} w_N^{m(j-N/2)} & m \text{ even} \\ \\ -w_N^{m(j-N/2)} & m \text{ odd} \end{array} \right. \qquad (4)$$

Both decimation-in-frequency, DIF, and decimation-in-time, DIT, FFT algorithms achieve the reduction in the number of arithmetic operations by recursively computing the DFT. An N-point DFT is computed from two N/2-point DFT's.

In our derivations the computations in (3) are instantiated in time and space into N-cycles and logN units, where logN denotes $\log_2 N$ throughout this paper.

In our DIF variant j is spread in time such that values of the input signal, X, enters in normal order. As a result of this order and design desicions the output is produced in bit-reversed order.

In our DIT variant m is spread in time such that values of the output signal, Y, are produced in normal order. As a result of this order and design decisions the input has to be provided in bit-reversed order.

The bit-reversed order results from the repetitive application of chosing even-before-odd.

Decimation-in-frequency, DIF

In the following derivation of a concurrent algorithm for the DIF type FFT we use the notation $[t]=t \bmod N$, and $\langle t\rangle = t-[t]$. Our derivation concerns the case where a sequence of DFT's are computed on contiguos windows of N samples, $N=2^n$, and the phase is chosen such that $k=\langle t\rangle=0,N,2N\ldots$.

With these definitions and assumptions equation (3) is rewritten as

$$y(m,\langle t\rangle) = \sum_{[t]=0}^{N-1} w^{m[t]}\, x(t) = \sum_{[t]=0}^{N-1} w^{m[t]}\, x(\langle t\rangle+[t]) \qquad m=0,1,\ldots,N-1 \quad (5)$$

The summation ($\sum$) in (5) is performed over time. There are N different sums (one for each m) the order of which is not specified by (5). No component of the DFT can be available until [t]=N-1.

Using (4) in (3) yields:

$$y(2m,\langle t\rangle) = \sum_{[t]=N/2}^{N-1} w^{2m[t]}\, (x(\langle t\rangle+[t]) + x(\langle t\rangle+[t-N/2]))$$

or

$$y(2m,\langle t\rangle) = \sum_{[t]=N/2}^{N-1} w^{2m[t]}\, (I + Z^{N/2})\, x(\langle t\rangle+[t]) \qquad m=0,1,\ldots,N/2-1 \quad (6)$$

and

$$y(2m+1,\langle t\rangle) = \sum_{[t]=N/2}^{N-1} w^{2m[t]}\, w^{[t]}\, (x(\langle t\rangle+[t]) - x(\langle t\rangle+[t-N/2]))$$

or

$$y(2m+1,\langle t\rangle) = \sum_{[t]=N/2}^{N-1} w^{2m[t]}\, w^{[t]}\, (I - Z^{N/2})\, x(\langle t\rangle+[t]) \quad m=0,1,\ldots,N/2-1 \qquad (7)$$

Expressions (6) and (7) still defines N summations, but they are partially ordered into two sets, the set of even and the set of odd components of the DFT. Each summation ($\sum$) is to be performed over time in the interval $N/2 \leq [t] < N$. The even components are computed from one intermediate signal, the odd components from another. Indeed, as is well known, [Cooley, Tukey 65], the even and odd components are simply DFT's of length N/2 on these intermediate signals.

In the following a subscript on a signal name, such as X_N, indicate that this signal is treated as a sequence of "windows" each of length N.

We chose to compute the even components of the DFT before the odd components. Hence, one network computing DFT's of size N/2 on the intermediate signals should be sufficient in order to obtain a DFT of size N in N cycles. No order between the components within a set is yet specified. With the assumptions made so far even components can be available from time [t]=N-1 and thereafter, and the odd components from time [t-N/2]=N-1.

The two intermediate signals of (6) and (7) are computed concurrently. A new intermediate signal, $X_{N/2}$, is defined. It consists of the intermediate signal for the even components followed by the intermediate signal for the odd components of the DFT delayed by N/2 cycles.

$$x_{N/2}(t) = (I + Z^{N/2})\, x_N(t) \qquad N/2 \leq [t] < N$$

and (8)

$$x_{N/2}(t) = Z^{N/2}\, \{w^{[t]}\, (I - Z^{N/2})\}\, x_N(t) \qquad N/2 \leq [t-N/2] < N$$

The selection mechanism implied by 8 can be implemented by a binary multiplexor controlled by a signal U_N defined as

$$u_N(t) = \begin{cases} 1 & N/2 \leq [t-N/2] < N \\ 0 & N/2 \leq [t] < N \end{cases} \qquad (9)$$

The intermediate signal $X_{N/2}$ can now be written as

$$x_{N/2}(t) = (u'_N(t)[I + Z^{N/2}] \,\#\, u_N(t)\, Z^{N/2}\, [w^{[t]}\, [I - Z^{N/2}]])x_N(t) \qquad (10)$$

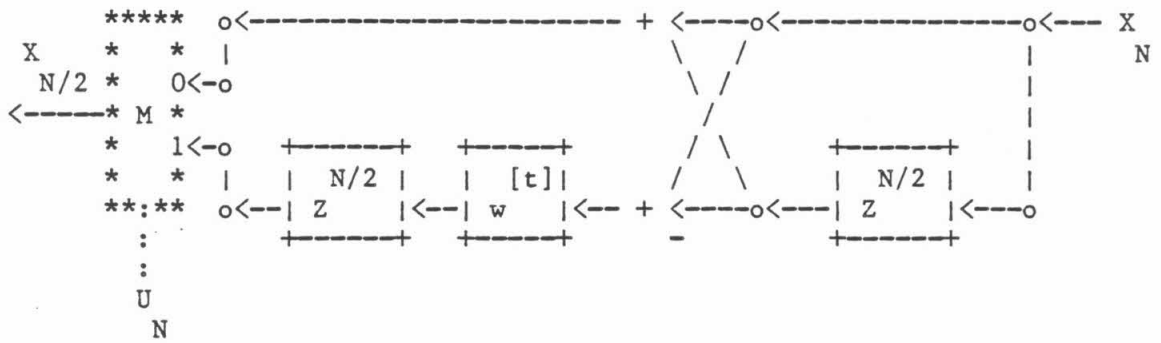

Figure 3: Computing $X_{N/2}$ from X_N

A network implementing (10) is shown in Figure 3.

By iterating the process that lead to (10) logN times all the components of y(*,<t>) are obtained in bit-reversed order. If DIF(N) denotes the operator that transforms the signal X_N into the signal $X_{N/2}$, then a complete DFT is performed by DIF(2)DIF(4)...DIF(N/2)DIF(N)X_N.

Each DIF(N) needs a control signal, U_N. The control signals for the different units are in phase with each other in the sense that when the signal with the lowest frequency has a 0-1 transition, so have all the others. Since it is easier to half the frequency of square waves than to double it, U_N is derived from $U_{N/2}$ unlike $X_{N/2}$ which is derived from X_N. Therefore, the control signal propagates in the opposite direction to the propagation of the data, X.

A simplified drawing of Figure 3 is shown in Figure 4, and an array for N=8 is shown in Figure 5.

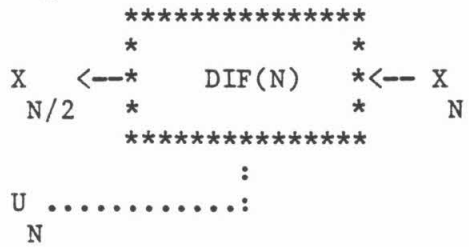

Figure 4: Computing $X_{N/2}$ from X_N

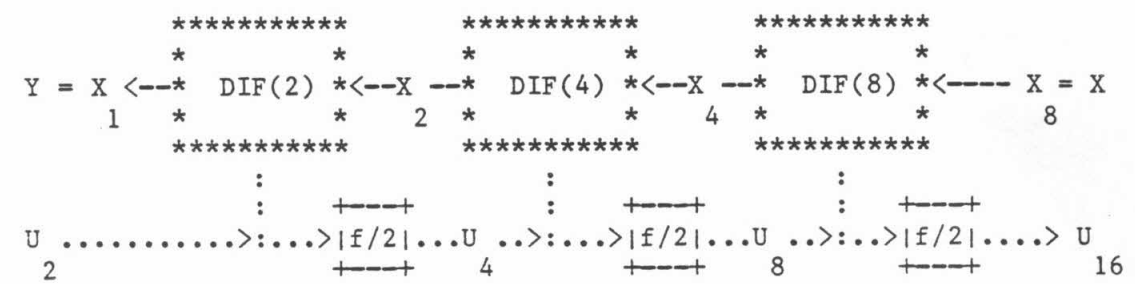

Figure 5: A decimation-in-frequency array for N=8

Decimation-in-time, DIT

In our derivation of space-time algorithms of the DIT type FFT we order the computations such that the components of the DFT are computed in order of increasing index.

$$y_N([t],\langle t\rangle) = \sum_{j=0}^{N-1} w^{[t]j} \, x_N(\langle t\rangle + j) \tag{11}$$

For DIT algorithms the summation in equation (11) is split into two, one containing even components of X, one containing the odd components.

$0 < [t] < N/2$

$$y_N([t],\langle t\rangle) = \sum_{j=0}^{N/2-1} w^{2[t]j} \, x_N(\langle t\rangle + 2j) + w^{[t]} \sum_{j=0}^{N/2-1} w^{2[t]j} \, x_N(\langle t\rangle + 2j+1) \tag{12}$$

$N/2 < [t] < N$

$$y_N([t],\langle t\rangle) = \sum_{j=0}^{N/2-1} w^{2[t-N/2]j} \, x_N(\langle t\rangle + 2j) - w^{[t-N/2]} \sum_{j=0}^{N/2-1} w^{2[t-N/2]j} \, x_N(\langle t\rangle + 2j+1) \tag{13}$$

By properly observing the properties of the coefficients, w, the time arguments in the right hand side of (13) have been made the same as those in (12). The two expressions define the DIT butterfly.

Each summation ($\sum$) in the above two expressions represent N/2 sums. We define a new signal, $Y_{N/2}$, that during the interval $0 \le [t] < N/2$ carries the sums based on the even components of X_N, and during the next half window carries the sums based on the odd components of X_N.

$0 < [t] < N/2$

$$y_{N/2}([t],\langle t\rangle) = \sum_{j=0}^{N/2-1} w^{2[t]j} \, x_N(\langle t\rangle + 2j)$$

$N/2 < [t] < N$

$$y_{N/2}([t],\langle t\rangle) = \sum_{j=0}^{N/2-1} w^{2[t]j} \, x_N(\langle t\rangle + 2j+1)$$

Hence, the two expressions (12) and (13) can be written as

$0 < [t-N/2] < N/2$

$$Z^{N/2}_{N}\, y_N([t],\langle t\rangle) = Z^{N/2}_{N/2}\, y_{N/2}([t],\langle t\rangle) + w^{[t]}_{N/2}\, y_{N/2}([t],\langle t\rangle)$$

or

$$Z^{N/2}_{N}\, y_N([t],\langle t\rangle) = (Z^{N/2}_{N/2} + w^{[t]}_{N/2} I)\, y_{N/2}([t],\langle t\rangle) \qquad (14)$$

$N/2 < [t] < N$

$$y_N([t],\langle t\rangle) = Z^{N/2}_{N/2}\, y_{N/2}([t],\langle t\rangle) - w^{[t]}_{N/2}\, y_{N/2}([t],\langle t\rangle)$$

or

$$y_N([t],\langle t\rangle) = (Z^{N/2}_{N/2} - w^{[t]}_{N/2} I)\, y_{N/2}([t],\langle t\rangle) \qquad (15)$$

For each instance of time two components of the DFT are computed according to (14) and (15). Even though it might be of limited interest to sequentialize the output, we will do so in order to arrive at a ("general") unit that has one input and one output.

A sequential output is obtained by delaying the high order half of the DFT components by N/2 steps before they are output. The selection between the streams of low and high order components of the DFT is made by a multiplexor controlled by a signal U_N, defined as in equation (9).

With these assumptions the two expressions for the output signal Y_N can be unified as:

$$Z^{N/2}_{N}\, y_N([t],\langle t\rangle) = (u'(t)[Z^{N/2}_{N} + w^{[t]}_{N/2} I] \# u_N(t) Z^{N/2}_{N/2}[Z^{N/2}_{N/2} - w^{[t]}_{N/2} I])\, y_{N/2}([t],\langle t\rangle) \qquad (16)$$

A network implementing (16) is shown in Figure 6.

The network in Figure 6 contains the same components as the network in Figure 3. The complexity and control of the output unit for the DIT algorithm is the same as the complexity and control of the input unit for the DIF algorithm. The difference between the units lies in the way the elements are interconnected.

If sums based on odd components are ordered before sums based on even components the delays in the upper path of Figure 6 would instead occur in the lower path after the multiplier. In that case the DIT butterfly

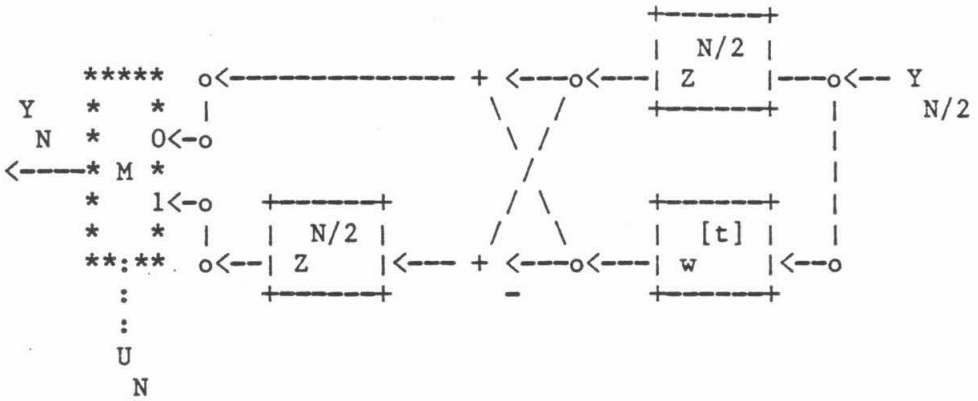

Figure 6: Computing Y_N from $Y_{N/2}$

unit would essentially be a mirrored version of the DIF butterfly unit. However, the input would have to be supplied in inverted bit-reversed order.

The inverse Fourier Transform is obtained by exchanging w with its inverse and normalizing. It is readily checked that if the units in Figure 6 and Figure 3 are connected in series and one unit use w^{-1} instead of w, then the input signal (multiplied by a factor of 2) is also the ouput signal, irrespective of which unit is used as input unit.

The procedure that resulted in (16) can be applied logN times. However, it should be noticed that for instance instead of an expression for $Y_{N/2}$ an expression for $Z^{N/4}Y_{N/2}$ is obtained. By applying $Z^{N/4}$ to both sides of (16) the expression for $Z^{N/4}Y_{N/2}$ can be used in (16). Hence, eventually an expression for $Z^{N-1}Y_N$ is obtained, i.e., our notation and derivations properly accounts for the fact that no frequency component can be computed before all required data is available. The first component of Y is available at [t]=N-1.

If DIT(N) is the operator that transforms $Y_{N/2}$ into Y_N, then a complete DFT is generated by DIT(N)DIT(N/2)...DIT(2)X.

A simplified drawing of Figure 6 is shown in Figure 7, depicting the operator DIT(N).

Figure 7: Computing Y_N from $Y_{N/2}$

An array for N=8 is shown in Figure 8.

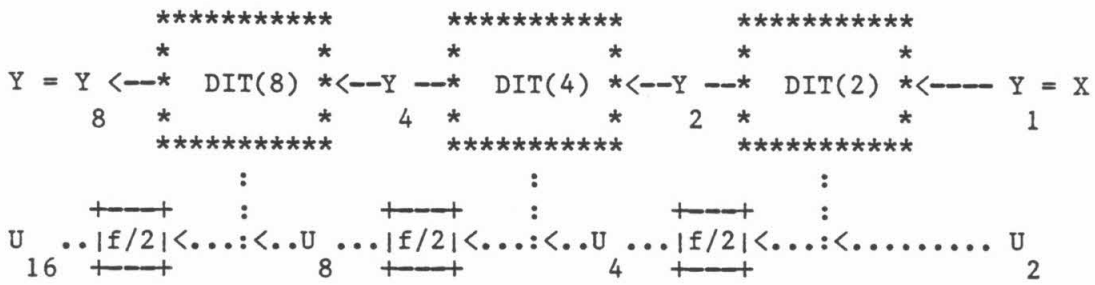

Figure 8: A decimation-in-time array for N=8

SUMMARY

In this paper concurrent FFT algorithms have been derived in a formal manner from the definition of the Discrete Fourier Transform and either an assumed order of the input data, or an assumed order of the output. With assumed input order the order of the output is a consequence of decisions of how to sequentialize the results of a butterfly operation. With assumed output order the order of the input data is affected by the same decisions.

Our implementations of FFT algorithms requires explicit control. The control is included in the expressions describing the computations of each unit in the array. The ratio of the frequencies of the control signal for neighboring units is two. Hence, the control signals can be obtained from the different stages of a binary counter. The ratio of frequencies of the control signals for the input unit to that for the output unit is 2/N for the DIF algorithm and N/2 for the DIT algorithm.

Different implementations of FFT algorithms can be obtained by formal transformations of expressions in our notation. The total storage requirement, the communication topology, the sequence of coefficients needed, and the control may vary.

The use of our notation suggests the use of shift registers in synchronous systems. It is strictly an ordering operator that also can be implemented in a self-timed system.

Expressions in the notation can be given an interpretation in the implementation domain, including sequencing and control.

ACKNOWLEDGEMENT

This work was sponsored by the Defense Advanced Project Agency (DARPA) under Contract No. MDA903-81-C-0335 with USC/Information Sciences Institute and Contract N00014-79-C-0597 with the California Institute of Technology. Views and conclusions contained in *this* paper are the author's and should not be interpreted as representing the official opinion or policy of DARPA, the U.S. Government, or any person or agency connected with them.

REFERENCES

[Cohen, Tyree 79]
Cohen D., Tyree V. C.
VLSI System for Synthetic Aperture Radar (SAR) Processing.
In Proceedings of SPIE, vol. 186, pages 166-177. Society for Photo-Optical Instrumentation Engineers, August, 1979.

[Cohen 78]
Cohen D.
Mathematical Approach to Iterative Computational Networks.
In Proceedings of the Fourth Symposium on Computer Arithmetic, pages 226-238. IEEE Computer Society, October, 1978.

[Cooley, Tukey 65]
Cooley J. W., Tukey J. W.
An Algorithm for the Machine Calculation of Complex Fourier Series.
Mathematics of Computation 19:297-301, 1965.

[Johnsson et. al. 81]
Johnsson, S. Lennart, Weiser, U., Cohen, D., and Davis, A.
Towards a Formal Treatment of VLSI Arrays.
In Proceedings of the Second Caltech Conference on VLSI, pages . Caltech Computer Science Department, January, 1981.

[Johnsson, Cohen 81a]
Johnsson S. L., Cohen D.
A Mathematical Approach to Modelling the Flow of Data and Control in Computational Networks.
In H. T. Kung, B. Sproull, G. Steele, editor, VLSI Systems and Computations, pages 213-225. Computer Sciences Press, October, 1981.

[Johnsson, Cohen 81b]
Johnsson S. L., Cohen D.
A VLSI Approach to Real-Time Computation Problems.
In Proceedings of SPIE, vol. 298. Society for Photo-Optical Instrumentation Engineers, August, 1981.

[Weiser, Davis 80]
Weiser, Uri and Davis, Alan L.
Mathematical Representation for VLSI Arrays.
Technical Report UUCS-80-111, University of Utah, Computer Science Department, September, 1980.

[Weiser, Davis 81]
Weiser Uri, Davis Al.
A Wavefront Notation Tool for VLSI Array Design.
In H. T. Kung, B. Sproull, G. Steele, editor, VLSI Systems and Computations, pages 226-234. Computer Sciences Press, October, 1981.